IBM WebSphere Message Broker vs. Microsoft BizTalk Server 2006

Greg Manship
WebSphere Americas Sales Executive Business Integration
manshipg@us.ibm.com

May 12, 2006

Table of Contents

Executive Overview:

IBM's WebSphere® Message Broker V6 provides a comprehensive set of capabilities that enable you to transform and route business data as it moves between applications, spanning multiple formats and protocols, easier, faster and more effectively than the BizTalk Server from Microsoft®. WebSphere Message Broker can transform message formats and content, enabling communication between applications that may have very different message formats and transport mechanisms and which were never expected to need to communicate with each other. Microsoft's BizTalk Server does not have the ability to support as many file formats and transport mechanisms as WebSphere Message Broker. Message Broker has the ability to perform message routing between applications, so that data can be distributed across applications without the need to change the sending applications, thus enabling you to add new applications much more easily. WebSphere Message Broker can natively route and transform messages in memory, providing superior performance gains over BizTalk Server, which must store each message that is to be routed in a Microsoft SQL Server database first, which effects performance. Message Broker has the ability to implement protocol conversion that enables disparate applications to be connected together; thus making it easy, for example, for a file-based application to communicate with one that uses WebSphere MQ or JMS messages. On the other hand, BizTalk Server is limited to support of the industry leading WebSphere MQ when it only resides on a Microsoft Windows Server, and provides no support for JMS messages.

Routing and transformation logic that you code in WebSphere Message Broker is held within WebSphere Message Broker and not in the applications, thus enabling you to easily achieve separation of concerns. The end-point applications contain the business logic that applies the business processing rules important to your enterprise, while operations such as protocol and message transformation or routing are contained solely within the WebSphere Message Broker environment, thus making your IT infrastructure much more flexible. Microsoft's BizTalk does not enable an SOA environment. It requires adapter support to reside only on the BizTalk Servers, and more commonly, most customers build their own adapters due to the lack of business functionality of the Microsoft adapters. WebSphere Message Broker has a robust design and scalable architecture, as well as a high level of performance, that Microsoft cannot match when it comes to scalability and performance. Customer implementations of Microsoft BizTalk Server find that they have to scale out the BizTalk and Microsoft SQL Server architecture to meet their business requirements, which requires more software licenses and hardware to purchase, as well as both software and hardware patches to support within their data center, thus increasing the total cost of ownership. WebSphere Message Broker, with all of its native characteristics combined with ease of use; provide the facilities you need to build an enterprise service bus (ESB) to implement an enterprise-wide service-oriented architecture (SOA).

What is BizTalk Server?

This is Microsoft's product for Business Integration, Enterprise Service Business and Business Process Management. BizTalk Server covers all of these areas to some degree, but it's their one answer to anything having to do with business integration. This is an important sales tactic that Microsoft will leverage within your customer install base -- one product for all of the customer's needs.

BizTalk has been around since 2000. They came up with a minor release in 2002. Those are both COM-based architectures. The revised product, re-released in 2004, was the first of the Microsoft server products to be re-written in the .NET technology. So it is fundamentally a .NET product. There's potentially some COM in some of the adaptors, but other than that the core product itself is a .NET product.

On November 7, 2005, Microsoft announced the release of BizTalk Server 2006, with general availability of the product in first quarter of 2006. Like its predecessors, this latest release allows connecting diverse software, then graphically creating and modifying process logic that uses that software. The product also lets information workers monitor running processes, interact with trading partners, and perform other business-oriented tasks. BizTalk Server 2006 should be considered a point release, and not a major version release. Built on the foundation of its predecessor, BizTalk Server 2004, this new release will look familiar to anyone who's used this earlier version. The most important new additions in BizTalk Server 2006 are:

- Better support for deploying, monitoring, and managing applications.
- Significantly simpler installation.
- Improved capabilities for Business Activity Monitoring (BAM).

BizTalk Server 2006 also uses the latest releases of other Microsoft technologies. It's built on version 2.0 of the .NET Framework, for example, and its developer tools are hosted in Visual Studio 2005. For storage, the product can use SQL Server 2005, the latest version of Microsoft's flagship database product, or SQL Server 2000, the previous release. BizTalk Server 2006 can also run on 64-bit Windows, taking advantage of the larger memory and other benefits this new generation of hardware offers.

It is important to understand some key concepts within BizTalk Server when competing against Microsoft.

Key Components of BizTalk Server 2006:

Orchestration within BizTalk Server:

One of they key components within BizTalk Server is the Orchestration capabilities, which allows a .NET developer in Visual Studio 2005 to design flow, interpret and generate data, call custom code, and organize it all in an intuitive visual drawing. BizTalk Server Messages that send and receive actions through ports, which it then uses as transported, are all fundamental elements of an orchestration. The message is the medium by which orchestrations communicate with the outside world and by which e-business is conducted.

Within BizTalk Server, there are Receive and Send shapes, which encapsulate the functionality that the .NET developer needs to receive messages into the orchestration and send messages from it. BizTalk accomplishes all of this in a logical graphical flow within Visual Studio 2005. In addition, BizTalk Server uses advanced orchestration concepts such as Web services, correlation, and long-running transactions.

BizTalk allows the .NET developer to define message types, ports, port types, and operations in a standard way, which allows disparate systems to communicate effectively with each other. BizTalk defines Correlation as the mechanism by which messages are associated with particular running instances of an orchestration, so that business processes gets the appropriate information when many instances are running and many messages are being sent back and forth.

Pipelines:

A pipeline is a piece of software infrastructure that contains a set of .NET or COM components that process messages in a predefined sequence. A pipeline divides processing into categories of work called stages, and determines the sequence in which the stages are performed. Each stage defines logical work groups, determines which components can go in that stage, and specifies how the pipeline components in the stage are run.

Within each stage, pipeline components perform specific tasks. For example, components within stages of a receive pipeline may decode, disassemble, and then convert documents from other formats to XML. Send pipelines do essentially the opposite: convert documents from XML to other formats, assemble, and encrypt, with each pipeline component performing a portion of the entire process. Although a stage is a container of components, each stage is itself a component with metadata. Stages have no execution code, as opposed to pipeline components, which do have execution code.

Ports:

In BizTalk Server, Ports specify how the orchestration will send messages to and receive messages from other business processes. Each port has a type, a direction, and a binding, which together determine the direction of communication, the pattern of communication, the location to or from which the message is sent or received, and how the communication takes place. A port binding is the configuration information that determines where and how a message will be sent or received. Depending on type, bindings might refer to physical locations, pipelines, or other orchestrations. Direct binding is not compliant with the standards of the Business Process Engineering Language for Web Services (BPEL4WS).

Hosts:

A host instance is the physical installation of a host in a BizTalk Server. An administrator uses Windows Management Instrumentation (WMI) or the BizTalk Administration console to install host instances. A BizTalk Server can support multiple host instances.

The BizTalk Host object represents a logical set of zero or more runtime processes in which the IT can deploy services, pipelines, and other artifacts. The Host object also represents a collection of runtime instances (zero or more) where the deployed items physically run. After the .NET developer creates a host, they then can add physical BizTalk servers (host instances) to the host. They cannot add a BizTalk server to the same host more than once. A single host instance can be added to multiple hosts.

Adapters and receive locations (including pipelines), and orchestrations are all contained in the BizTalk hosts and perform the following functions:

- Receiving: This is where BizTalk does the initial processing of messages after they are picked up in a receive location.
- Sending: These items do the final processing of messages before they are sent out to the send port.
- Processing: These items process messages based on the instructions in an orchestration.

Microsoft recommends creation of different hosts for each function to create security boundaries and facilitate management. In reality, this creates an additional layer of management of the BizTalk architecture because of all the different hosts for processing and for receive/send, as well as having you separate trusted and non-trusted items.

Performance:

IBM Message Broker, when optimized, performs anywhere from 43.6 times to 90.8 times faster than the optimized Microsoft implementations of BizTalk Server in lab tests. BizTalk Server was natively designed for application integration, not routing and transformation. All documents that come into BizTalk Server get stored in the BizTalk Server SQL Server Message Box, and converted internally to XML by BizTalk. When a flat non-XML file comes into BizTalk Server, BizTalk must internally place xml tags associated with each line of data. As a result, this causes BizTalk Server to do roundtrips to the SQL Server database to store each object, which in return, causes dehydration in the BizTalk Server. As an example, if a simple text file is sent into the BizTalk Orchestration, it will not get routed until all lines are included with the xml tags. Finally, it is sent through a port after the entire message is stored in the BizTalk Server Message Box database.

BizTalk Server performance is affected by the message size, streaming data over the network, and schema complexity. Certain flat files, or what are known as deep noids for BizTalk Server, will cause parsing problems within BizTalk Servers Orchestration. The BizTalk Mapper will cause problems due to loading data into the DOM (Document Object Model) along with any custom functions.

In previous releases of BizTalk Server, mapping of documents always occurred in-memory. For BizTalk Server, in-memory mapping of data will quickly eat up resources when large documents are mapped. Therefore, in BizTalk Server 2006, Microsoft introduced support for large message transformations, which in IBM Message Broker has natively been part of the solution for years. A different transformation engine is used when transforming large messages so that memory is utilized in an efficient manner according to Microsoft. When dealing with large messages, the message data is buffered to the file system instead of being loaded into memory using the DOM (Document Object Model), these messages still need to then go from the file system to the message box before being routed accordingly. BizTalk does this so that memory consumption remains flat, as memory is used only to store the cashed data and indexes for the buffer.

However as the file system is used there is expected performance degradation when comparing with in-memory transformation, which is already well below IBM Message Broker in memory performance. Because of the potential performance impact, the two transformation engines will co-exist in BizTalk Server 2006.

So now we look at some of the in memory performance degrading in BizTalk Server. BizTalk Server is affected by any serialization with batched messaging (looping in a stream) and persistence will degrade performance due to every send, execute, end of transaction. This serializes the state and as a result dehydration takes place and an instance out of memory normally occurs. Within the BizTalk Server architecture, parallel actions are the slowest because each parallel operation is serialized within BizTalk. In addition, since messages in real time batching are all processed serial, BizTalk Server will incur additional performance issues and multiple roundtrips to the BizTalk Server SQL Server databases. The databases for BizTalk Server will compete for resources, which causes the architecture to require additional Windows Servers so that databases are separated out based on adapters, messages and host instances. Since the adapters and orchestrations are on separate host servers, this will require additional Windows Servers for the customer to patch as well as additional BizTalk Server licenses.

BizTalk Server has many areas to be concerned with regarding performance from CPU, Disk usage, and there is heavy disk usage on OLTP and for message database, serialization of messages, and finally Network performance. It is very common to see orchestrations that are not completing fast enough and as has been established, this is typically due to memory contention or CPU saturation. Although the .NET developer will attempt to simply the orchestration of the workflow because of the complexity into multiple smaller orchestrations, in reality these multiple workflows can easily cause additional latency and added complexity.

Message Box in BizTalk

What is the purpose of the Master Message Box database that BizTalk Server 2006 requires? It contains the state of orchestrations that are currently in progress. It is defined as the heart of the publish/subscribe engine in Microsoft BizTalk Server. The Message Box is made up of two components: one or more Microsoft SQL Server databases and the Messaging Agent. The SQL Server database provides the persistence store for many things including messages, message parts, message properties, subscriptions, orchestration state, tracking data, host queues for routing, and others. The BizTalk Server group may have one or more Message Box databases into which it publishes messages and from which subscribers to those messages extract messages.

The database provides some of the logic related to routing messages and fulfilling subscriptions. The Message Agent, however, is the component that encapsulates and abstracts the database component and is the interface used by BizTalk Server to interact with the Message Box. The Message Agent is a Component Object Model (COM) component that provides interfaces for publishing messages, subscribing to messages, retrieving messages, and so on. This interface is the only mechanism used by other BizTalk Server components, including the adapter framework and orchestrations, to interact with the Message Box. As a result, we see first off that Microsoft positions BizTalk Server 2006 as completely .NET, but the heart of the application is still COM, which in return affects performance.

If an orchestration fails there is no way to tell exactly what data has been lost from the Message Box databases, there are some steps the .NET developer can take to gather information about the lost data.

Debugging lost orchestrations is not a trivial task for the .NET developer; they will need to determine what messages have been sent and received in the current orchestrations, and what external systems have been used after the point of recovery. For example, if the system maintains an external log of messages and events, they can examine that log. The .NET developer may also need to manually review the external systems to see what activities have occurred.

After the .NET developer has been able to determine the cause of data loss, they then can begin to correct the restored system, deciding which processes can continue, which processes must be terminated and restarted (by resubmitting the lost activation messages), and which processes have completed successfully and can be terminated. This process depends largely on the architecture of your system and considered as part of your system recovery planning.

Again, the master Message Box performs the actual routing, which becomes very CPU intensive and since most customers will have more than one Message Box Microsoft has now added an additional layer of management by forcing the BizTalk solution to use the SQL Server Log Shipping capabilities to balance all the Message Box databases. Databases in the BizTalk architecture will always compete for resources, as a result the architecture expends due to the need to separate databases out based on adapters, messages and host instances. Finally, the solution requires having the Microsoft SQL Server Agent running, if not the BizTalk databases will grow and impact system memory.

Tracking Bottlenecks:

BizTalk Server has the ability to track host instances that will move both BAM and HAT data from the Message Box database to a Microsoft SQL Server Analysis Services table for business activity monitoring. If multiple Message Box databases are configured the tracking host instance uses four threads per Message Box. As a result, this capability now becomes another potential bottleneck that causes this host instance to consume additional CPU. Microsoft typically recommends scaling out this solution, requiring additional software licenses and hardware to manage and maintain.

It is very common that the Tracking Data table in the Message Box database will start backing up, usually because the data maintenance jobs on the database are not running as configured, causing growth on the databases. Once

these databases grow too large it can have a negative impact on the ability of the tracking host to insert data into these tables causing the tracked data to backup in the Message Box database tables and effect performance of the Message Box database in greater magnitude.

High Availability:

In BizTalk Server, you will have to have multiple message boxes configured; the first SQL Server database instance always becomes the Master Message Box. Now the BizTalk engine balances the load in a round robin fashion. However, each message subscriptions are held and matched only in the Master Message Box. In this case, this single Message Box will potentially be a single point of failure. If this box goes down, or database instance goes down, then BizTalk will not be able to do any subscription matching. The only high availability support that Microsoft can offer for the BizTalk Server Master Message Box database instances is to leverage the SQL Server 2005 Log Shipping. So as a result, the customer may setup multiple message boxes within MSFT SQL Server for BizTalk, but this is not high availability, it only leverages SQL Server Log Shipping, which is a file transfer of data from one database instance to a second database instance over a network. Because the BizTalk Server solution requires multiple SQL Server databases, if you have a single database server then that server should be clustered. That means that all databases on the BizTalk solution require clustering, even non-critical BizTalk databases. If there is a master database server for the routing message box and then it is recommended that content message database boxes be on separate SQL Servers, then each one of those SQL Servers databases must be clustered. If the BizTalk Management, Single Sign On (SSO), or any other database is on a separate SQL Servers, then those servers must be clustered as well. This dramatically increases the cost of the Microsoft solution, as well as the amount of servers, and database servers that need to be maintain and patched.

The BizTalk Servers do not lend themselves to running on a Cluster and there are many issues that cannot be readily resolved if BizTalk is on a clustered server, so if you use clustering to provide high availability for SQL Server then you must have a separate servers for BizTalk. Once you have a separate server for BizTalk then you must use two servers for BizTalk so that a single BizTalk server failure does not result in a single point of failure for the system to go down.

Because the BizTalk Server inherently stores all messages in Microsoft SQL Server, the SQL Server must use a SAN. The customer will need to put tracking and message box databases on different disks if they are on the same database machine.

Memory Starvation

High throughput scenarios can have increased demand on system memory. Since a 32-bit process is limited by the amount of memory it can consume, Microsoft recommends that the BizTalk architecture have separate the Receive/Process/Send functionality into separate Host Instances such that each host receives its own 2GB address space. The areas that affect BizTalk Server and cause memory starvation that are expensive costs are singleton, and atomic scope. Serialization with batched messaging, which is defined as looping in a stream, as well as persistence messages hurt performance within BizTalk Server. Orchestration such as long running transactions that become serialized because of every send, execute, and end of transaction will cause a performance hit and slow down BizTalk Servers. The reason for this is that BizTalk serializes the state, which becomes expensive internally to BizTalk especially when custom components are added.

If BizTalk Server 2006 is used for batching within an orchestration, it becomes an atomic scope, causing the messages in the batch to be processed serially and degrading performance. It is generally recommended to use short descriptive schema node names, because long names affect size of message, parsing, memory usage, and overall performance. If there are requirements around adding additional properties to the message scheme, BizTalk will attempt to only use promoting properties for routing of the message, because the more properties that are adding to the message the more performance will be impacted. If the .NET developer wants to track a property that has been promoted in a message, again this impacts performance on the BizTalk Solution.

Any time there is delivery notification, which is actually conducted by the Message Box agent, this impacts performance. It is important to remember that this delivery notification is coming from a SQL Server database that then gets routed through a Port that is running on top of a BizTalk Server host.

When a .NET developer creates parallel actions in the BizTalk application, these are the slowest due to the multiple states that incur within the BizTalk Mapper and are not parallel threads. Each parallel operation becomes serialized within BizTalk Server.

Will .NET 1.x assemblies migrate to 64-bit without a recompile?

No. Assemblies compiled with the .NET Framework 1.x will only run as 32-bit executables on the .NET Framework 2.0. This is a rule enforced by the .NET Framework 2.0 common language runtime (CLR). Therefore, customers are required to recompile their assemblies using the .NET Framework 2.0 and set the Any CPU flag to support both 32-bit and 64-bit CLRs. Customers that want to migrate BizTalk Server 2004 32-bit applications to BizTalk Server 2006 64 bit will incur an additional migration impact.

Message Modeling

WebSphere Message Broker provides the flexibility to transform ANY format to ANY format and this is all done natively without the needs for custom development. Microsoft's BizTalk Server 2006 does not provide the necessary transaction integrity out of the box that Message Broker is able to achieve. The combination of MQ and Message Broker provide built-in compensation capabilities. In the event that an update to a back-end enterprise system (e.g. database) needs to be rolled back or reverted to a previous state, the transactional capabilities inherent in a MQ / Message Broker solution provide that capability without custom development. BizTalk Server requires customization by the development team to achieve the same level of integrity, if there are multiple update or delete targets (Oracle, DB2, and SQL Server) and a message on an MQ message queue, then transaction integrity becomes important. For example, if the delete/insert failed on DB2, then the same delete/insert needs to be rolled back on Oracle, DB2, SQL Server, and the message on the message queue. Microsoft does not support this functionality natively out of the box, it requires developer customization.

Many integration initiatives require EBCDIC support. Message Broker provides the capability to accept ASCII and convert to EBCDIC in-flight prior to delivering to receiving system. This capability is provided out of the box with no customization required where as the BizTalk developer has to customize the orchestration to handle this conversion in-flight.

When it is necessary to provide modeling of a message that is a native flat file, Message Broker inherently provides support to process (i.e. parse, build, etc) flat files without the need for customization. Microsoft BizTalk Server does not support this functionality, but can with a meta-layer development requiring additional developer effort and many lines of code to achieve the same out of the box experience provided within Message Broker.

When the Message Broker developer is working with messages, they have complete flexibility to work with failed messages and make them resume-able, with BizTalk Server, this requires customization, and it is not easily achieved within BizTalk that creates additional challenges for the BizTalk architecture. When the BizTalk Developer is working with an inbound message, and they choose to dynamically make changes to this configuration, this becomes a challenge for the developer. If the Inbound configuration is for an array of web services support, the Microsoft solution will not be able to easily publish and consume the service without customization by the BizTalk developer. Both ESQL and Java handle arrays natively.

PUB/SUB Support:

The Microsoft BizTalk Server support in a publish/subscribe design is a limited topology with three components:

- Publishers
- Subscribers
- Events

Publishers include receive ports that publish messages that arrive in their receive locations, orchestrations that publish messages when sending messages or starting another orchestration asynchronously, and solicit/response send ports that publish messages when they receive a response from the target application or transport.

After a subscription is created and enabled, a message must be published before any processing takes place. Messages are published when they are received into BizTalk Server from one of the services mentioned previously. For this discussion of routing, we will focus on messages received into BizTalk Server through an adapter. When messages go through the receive pipeline processing, properties are promoted into the message context. After a message is ready to be published into the Message Box after being processed by the receive adapter and pipeline, the first thing that happens is the Message Agent inserts the property values for the promoted properties and predicate values from the message context into the master Message Box SQL Server database. Having these values in the database enables the Message Agent to make routing decisions. In other words, BizTalk Server has no way to natively route the messages, and determine ahead of time where a message should be routed.

Before the message can be routed, BizTalk Server still has additional steps, now the Message Agent of BizTalk has to ask the master Message Box database to find subscriptions for the current batch of messages being published. Keep in mind that stored procedures now have to query the Message Box for the subscription, find the predicate tables that have been identified, and link them to the message properties stored for the current batch of messages.

Now that the Message Agent has the information, it inserts the message once into each Message Box database for each subscription. From here stored procedure calls pass through the Message Box and look up the subscription detail information to have the message match application properties before sending the message into the application specific queue or application specific suspended queue depending on the state.

Compare this to WebSphere Message Brokers support that comes natively out of the box for a pub/sub architecture. Message Broker leverages collectives, which provide the following benefits:

- Messages destined for a specific broker in the same collective is transported directly to that broker and do not need to pass through an intermediate broker. This improves broker performance and optimizes inter-broker publish/subscribe traffic, in comparison with a hierarchical tree configuration. Microsoft on the other hand is requiring the Message Box to find subscriptions, query multiple Message Boxes that store the content and determine routing decisions.
- With Message Broker, if the clients are geographically dispersed, you can set up a collective in each location, and connect the collectives (by joining a single broker in each collective) to optimize the flow of publications and subscription registrations through the network. In theory, Microsoft can accomplish the same, but at the same time, there is a large network impact.
- Message Broker allows grouping of clients according to the shared topics that they publish and to which they subscribe. Clients that share common topics can connect to brokers within a collective. The common publications are transported efficiently within the collective, because they pass through only brokers that have at least one client with an interest in those common topics. Within the Microsoft solution, common content can be routed and stored in the allocated BizTalk Server Message Box, but the master Message Box is still going to have to query across the network to find this content before sending the data to the clients. A client can connect to its nearest broker, to improve its own performance. The broker receives all messages that match the subscription registration of the client from all brokers within the collective. The performance of a client application is also improved for other services that are requested from this broker, or from this broker's queue manager. A client application can use both publish/subscribe and point-to-point messaging. BizTalk Server does not support this type of architecture natively out of the box; this logic would have to be built into the solution.

In the BizTalk Server architecture, incoming messages have to go into the receive pipeline processing, and then properties are promoted into the message context. After a message is ready to be published into the Message Box after being processed by the receive adapter and pipeline, the first thing that happens is the Message Agent inserts the property values for the promoted properties and predicate values from the message context into the master Message Box SQL Server database. Having these values in the database enables the Message Agent to make routing decisions.

The next step is for the Message Agent to ask the master Message Box database to find subscriptions for the current batch of messages being published. Keep in mind that the messages have not yet been written to the database, only the properties from the context. Now BizTalk Server has to use stored procedure in the Message Box which then queries the subscription and predicate tables identified above, linking them to the message properties stored for the current batch of messages.

FTP Support:

Microsoft provides an FTP adapter with BizTalk Server 2006. One of the areas where Microsoft will position integration into the mainframe or an miseries system is via the FTP adapter, and this will be positioned as an inexpensive but Service Oriented Architecture approach. The BizTalk Server FTP receive adapter does not support receiving files from a partitioned data set on the miseries™ or mainframe. In addition, the adapter will not allow the use of the temporary folder function when sending files to an IBM or AS400 host. Any input into this field will be ignored.

Within the BizTalk solution using the FTP adapter, there is the possibility of data duplication or lost data. Possible Scenarios for Data Duplication or Data Loss When Receiving Data with the FTP Adapter are when the BizTalk Server FTP adapter uses the FTP client protocol to poll the designated FTP server and retrieves data from the server as is. The FTP adapter does not perform any validation of the data that it retrieves; it simply hands the retrieved document to the BizTalk Messaging Engine for processing and then deletes the original document from the FTP server. If the FTP adapter retrieves a document from the FTP server that is still being written to by the host application, then the retrieved document will be incomplete. If the FTP adapter retrieves an incomplete copy of the original document, data duplication or data loss can occur. This can happen when the original document is still being written to the FTP server by the host application, because then the FTP adapter will not be able to delete the document and will retrieve another copy of the document at the next polling interval configured for the receive location. This causes document duplication to occur. In addition, when the host application has finished writing the document to the FTP server, then the document will be deleted. This will cause data loss to occur. Microsoft cannot ensure the successful transformation of data with the use of their FTP adapter for BizTalk Server.

Complex Integration:

Microsoft consistently talks to customers about the flexibility within the .NET architecture to integrate and extend CICS applications into an SOA world. When in reality, Microsoft provides limited ability within their application suite to accomplish this as a service. Microsoft has the ability through one of two ways to integrate with a CICS application. The first is via WebSphere MQ. This is accomplished by leveraging the BizTalk Server MQ adapter, which will integrate messages from BizTalk Server to WebSphere MQ only when MQ runs on a Windows platform. Microsoft BizTalk Adapter for MQSeries has two parts: the adapter running under BizTalk Server 2004 and a COM+ component, MQSAgent, running under MQSeries Server for Windows. By having the customer attempt to integrate with the CICS application the customer is limited to only being able to send and receive messages via MQ Server, or distributed transactions support between MQ and the mainframe (only on the Windows operating system), and allow the developer to only have access to the MQ header properties.

For more extensive integration needs, Microsoft will leverage Host Integration Server (HIS) 2004. HIS 2004 is used to connect and integrate applications, data sources, messaging, and security systems on IBM mainframes with Microsoft Windows environments, enabling the re-use of host assets across distributed networks.

While HIS 2004 provides access to CICS, it merely builds a .Net proxy for the CICS target system. When the BizTalk developer leverages HIS 2004 to connect to the CICS system, they can only import the COBOL into Visual Studio. From here the BizTalk developer must build the schema from the imported COBOL, which impedes on the concept of ease of use for the Visual Studio developer, in addition the existing schema cannot be referenced from the BizTalk Orchestration files. As a result, the developer must decide ahead of time which fields they want to reference. In the BizTalk application, the Microsoft developer typical creates message variables for interacting with different systems i.e. SAP and a Web Service. However, message variables are not a service, and instead the developer must use regular variables for interacting with COBOL. The impact for the developer is that now there is no service interaction, it is handled by explicitly calling the proxy object that was created with Host Integration Server that is a .NET object.

The architecture of the WebSphere® Message Broker message flow allows for the interface between a CICS application and a UNIX® web service to be consumed and implemented in a consistent single interface that adheres to an SOA approach. The basic solution architecture, with a single web service is based on the capability of the ESQL “Propagate to Terminal …” function in combination with the ability to retain information saved in the environment on different paths and access that information in a consistent manor. Message Broker in return provides the ability unlike a BizTalk Server application, to have the inbound message id from COBOL be used to create the outbound correlation id, as well as the ability to logically “group” multiple messages. This is done by Message Broker using a unique correlation ID for each group of messages.

It is possible for BizTalk Server to interact with CICS applications and expose COBOL as a web service, as well as the file system of non-Windows platforms, but not as a service within the SOA architecture. Each non-Windows platform application cannot be consumed as a service and allow the BizTalk Developer to leverage a single development approach.

Adapters:

Microsoft positions that they have over 350 adapters, and that they have more adapters than any other vendor in the industry. Although this is a true statement, it is important to understand the truth behind this statement. To date Microsoft actually owns less than 20 adapters. In August of 2005, Microsoft announced the purchase of eight new .NET adapters from IWAY Corporation. These eight new adapters are only supported by BizTalk Server 2006. The additional adapters are all third party adapters with the majority of them being COM based.

A fundamental difference between WebSphere Message Broker and the BizTalk Server adaptors are in the process phase of BizTalk. Message Broker has configuration flexibility that allows the adapter to run on the same box as the Message Broker server, but it does not run in the same process space of the server so there's better isolation from the server. Microsoft's BizTalk Server can run on separate host within the BizTalk Server, but this does share the same process and is unable to be isolated from other BizTalk orchestrations. As a result, the BizTalk adapters effect performance, and if a BizTalk adapter causes a failure, it has the potential to bring down the other BizTalk Server hosts. This is one main reason why Microsoft recommends the adapters be configured on separate BizTalk Servers to prevent performance bottlenecks and disruption of other orchestrations sharing the same processes. IBM provides the communication between the adaptor and the server to be done via JMS messaging which provides for a more flexible architecture.

WebSphere Message Broker has the advantage of allowing a run time adaptor to be placed on the same system that the application or the technology which in return provides for reliable messaging between the application and the adapter. Microsoft cannot achieve this flexible of an architecture, all BizTalk adapters must reside on the actual BizTalk Servers. Since Microsoft recommends that BizTalk Server adapters reside on their own separate host within a BizTalk architecture, more times than not, this drives up costs as the customer now has to purchase additional BizTalk Server licenses along with Windows Server licenses.

Cluster support for high availability in the BizTalk Server adapter still only provides provide high availability for running a single instance of an adapter for purposes of ordered delivery. All of the BizTalk Adapter handlers with the exception of the EDI Adapter can be run in a clustered host but there is no benefit derived for running adapter handlers in a clustered host in the Microsoft architecture. The BizTalk adapters when running in a clustered host instance effectively reduces the throughput for the adapter handler by half because the clustered host only runs on one cluster node at a time. In addition the Microsoft MSMQ Adapter receive handler does not support remote transactional reads, only local transactional reads are supported. The MSMQ adapter receive handler must run in a host instance that is local to the clustered MSMQ service in order to complete local transactional reads with the MSMQ adapter. In addition, the EDI adapter support on the Microsoft architecture cannot run in a clustered BizTalk Server environment, and the majority of the time Microsoft recommends to customers that they go with the Covast EDI adapter. This again increases costs and complexity to the Microsoft solution.

Deployment:

Microsoft has reduced the deployment efforts in BizTalk Server 2006, but there are still complexities that IT must face when deploying and redeploying versions of the BizTalk Server application. Microsoft BizTalk Server and the .NET Framework, and all BizTalk Server artifacts (maps, schemas, orchestrations, and pipelines) get compiled into what is known as .NET assemblies. If a BizTalk assembly in an existing application is updated, the developer may need to restart host instances for the changes to take effect. Restarting a host instance stops all other applications that are running on the host instance.

If the BizTalk developer chooses to use Visual Studio to deploy an application into a production environment, all BizTalk instances will have to be restarted when the developer deploys the application, including those that are not associated with this instance of the BizTalk application. This will stop all other applications that are running on any host instance on the local computer.

When the BizTalk developer revises assemblies that contain an orchestration, schema, or map, they must update the global assembly cache (GAC) with the assembly containing the new version. Otherwise, BizTalk Server will use the out-dated version. To do this, on each computer that is running an instance of the host to which an application is bound, the developer now has to go and uninstall from the GAC the outdated version of the assembly containing the updated artifact and make sure that the new version is installed.

If the BizTalk developer deploys an entirely new application to replace an existing application, they then have to correct any references between other applications and the application that they are replacing. In addition, the use of .NET policy files is not supported in BizTalk Server 2006. This is because a policy file may not work as expected. Policy files redirect .NET to a specified assembly version in the GAC, but BizTalk Server often accesses assemblies and artifact data from a cache or the BizTalk Management database. Depending on the artifact type, caching situation, and whether host instances are restarted, the policy file may not do what is desired.

The BizTalk developer can also encounter troubles with assemblies that are added to an application that includes a new schema with the same message type as an existing schema in the BizTalk group, the schema with the highest version number will be used when schema resolution occurs in pipelines. In addition, if one message type refers to more than one .NET type, this ambiguity may cause pipeline execution failure. By deploying new assemblies, the BizTalk solution can have an impact on pipeline, which means BizTalk can potentially have delivery failure of a message. In addition, if the BizTalk developer chooses to un-deploy the latest schema it will cause the previous version of the schema to automatically become available, if that schema is still present. The net result is that IT and the .NET developer have to deal with a complex architecture and ensure at all times what is the correct BizTalk assemblies being deployed, if not there is the chance of message failure and version incompatibility.

It is very common for the BizTalk developer to build an orchestration with custom components. When the developer now has to successfully assemble and deploy this application and run it, they often will receive errors such as "File or Assembly name or one of its dependences not found." This error means that the BizTalk developer now has to ensure that BizTalk Server can find each custom component that was built, and manually assemble that in the BizTalk application in the global assembly cache of the computer that hosts the application, because the BizTalk Server deployment wizards will not assemble the custom components as part of the BizTalk application, as a result increasing the amount of time and effort spent to deploy a BizTalk solution.

Web Services Limitations in BizTalk Server 2006:

Although BizTalk Server 2006 has strong web services support, there are still many areas that add an additional layer of complexity for the developer. Web Service interactions are handled differently than other types of interactions. They are not based on schemas in the BizTalk project, but instead are based on the message definitions inside a WSDL file. Web Services in BizTalk Server 2006 do not adhere to the same property and default values as the .NET 2.0 framework. What does this mean, now the .NET developer has to spend additional developer effort managing web services generated by BizTalk Server because Microsoft is not adhering to their own properties within the .NET framework that BizTalk Server is suppose to be built upon and leverage. As an example of the impact to a solution built on top of BizTalk Server 2006, BizTalk Server uses a different web services time out value than the .NET 2.0 Framework. So if a web clients that is using the .NET 2.0 framework is calling a Web Service generated with the BizTalk Server 2006 Web Services Publishing Wizard, it is possible that the client cannot receive server script timeout errors because the client request timeout occurs first by default.

Typical web services attributes that the .NET developer is familiar within the .NET 2.0 framework for customization are not customizable in BizTalk Server 2006. This is because BizTalk's Web Services Publishing Wizard does not allow the .NET developer to customize the Web service or Web method. The publishing wizard automatically generates attributes that are set based upon information that the wizard provides and does not provide the flexibility for the developer to customize these web services. In addition, if the developer attempts to modify any attribute that the BizTalk Web Services Publishing Wizard generates, this may cause the Web service to function incorrectly.

In BizTalk Server 2006 there is the Web Services Publishing Wizard, but when publishing web service schemas and using such properties as the include element in the schemas, BizTalk will not be able to publish the web service. An error will occur for the BizTalk developer. The publishing wizard for web services that comes with BizTalk Server 2006 has limitations around XSD.exe that are also part of the same limitations in the 2.0 .NET framework.

Another example of challenges that the BizTalk developer encounters is when publishing web services that have schemas that specify values for minOccurs or maxOccurs attributes, these values may be different in the schema when published as a web service. Therefore, the BizTalk developer in the application defines values within the web service, but then at run time the attributes of those values change, requiring the developer to manual have to go in an change values. Many times inside of BizTalk Server when the developer chooses to use published web service schemas that it will the developer to have to manually modify the generated web project and also have to restart IIS, which means bringing down the BizTalk Server due to previous processes that need to be cleared out within IIS cache.

The BizTalk Web Services Publishing Wizard allows the developer to republish a published Web service but it will not store any previous web service settings, if the .NET developer makes a mistake or want to use that web service in the future, they will not be able to in BizTalk Server.

BizTalk uses the .NET XSLT engine, which does not support XSLT 2.0 (not even on .NET 2.0). However, you can use script and custom functoids in your BizTalk Maps to do these kind of things.

Summary:

IBM WebSphere (R) Message Broker Version 6.0 delivers the solution for your application integration and information mediation needs today. It can add to the capabilities of your enterprise service bus (ESB) by extending the integration of your service-oriented architecture (SOA) to encompass your entire enterprise and beyond. WebSphere Message Broker Version 6.0 adds real value to the reliable connections between applications and systems as provided by WebSphere MQ and other WebSphere products. IBM's WebSphere Message Broker V6 provides superior performance and scalability to meet a customer's enterprise business integration requirements. WebSphere Message Broker minimizes complexity in integration by providing a single solution that provides a superior solution against Microsoft BizTalk from installation and configuration through message flow development, administration, run-time management, and performance. These changes not only simplify installation and configuration, but also provide more facilities that enable you to extend the impact of WebSphere Message Broker in your enterprise. In addition to functional improvements over BizTalk Server 2006, performance is also significantly greater in key areas such as message parsing and writing, and ESQL processing. These functionalities reduce the processing cost of running message flows, as well as the number of software and hardware licenses.

Microsoft's BizTalk Server 2006 is an incomplete solution that does not provide as much rich functionality to address customer's current and future requirements. There is a decrease in value for BizTalk customers due to the lack of supported adapters that in general require customers to build their own adapters due to the lack of functional business need of the current offering of adapters. BizTalk Server 2006 lacks a consistent model to connect to platforms; a developer will connect to an SAP system differently than an AS/400 or zSeries platform. The BizTalk Solution increases the total cost of ownership based on requiring Microsoft SQL Server, Host Integration Server and third party products to complete the solution offering.

WebSphere Message Broker's robust design, scalable architecture, high performance, and ease of use deliver the capability to build your ESB to implement an enterprise-wide SOA in stages. This flexible combination of capabilities provides your business with the tools to solve your integration requirements from any starting point: a simple two-application solution or a multi-application, multi-platform design. As your business expands, and the integration challenges multiply, WebSphere Message Broker can accommodate the changing needs of your business by growing your infrastructure without increasing the complexity, protecting your existing and ongoing investments in applications and data structures and seamlessly extending your enterprise service bus. IBM WebSphere Message Broker is the cost-effective solution for your enterprise's extensive integration and mediation needs today and will protect your investment into the future.